Selenium Java Project Code Walkthrough by Mariam El-Ghazy

1. Project Overview

This project automates browser interactions using Selenium WebDriver, organized with:
- Maven as the build tool (pom.xml)
- Cucumber for BDD-style testing (sample: login.feature)
- Page Object Model (POM) for modular page interaction
- JUnit/TestNG for test execution

2. Project Structure

```
Selenium_JavaProject/
├── pom.xml                    # Maven dependencies & project config
├── resources/
│   └── chromedriver.exe        # WebDriver
├── src/
│   ├── features/
│   │   └── login.feature       # Gherkin scenario(Cucumber)
│   ├── main/java/Pages/        # Page Object Model classes

│   └── test/java/
```

# Base setup/teardown for tests

#Login Test(Sign-in , Edit and filtering test cases)

#runners for Cucumber Test Runner

#Step definitions for Cucumber

3. Key Components

pom.xml

Contains dependencies such as:
- selenium-java
- cucumber-java, cucumber-junit
- junit or testng
Defines Maven's build lifecycle and plugins.

Feature File: login.feature

Feature in Gherkin syntax:

```
Feature: Login functionality

  Scenario: Valid login
    Given I am on the login page
    When I enter valid credentials
    Then I should be redirected to the account page
```

Page Object Classes (Pages/)

```
src/main/java/Pages/

├──── AccountPage.java

├──── HairCarePage.java

├──── HomePage.java

├──── LoginPage.java

└──── SecureAreaPage.java
```

The **Page Object Model** is a design pattern in Selenium that promotes better test maintenance and code readability. Each page is represented by a **Java class**, which contains:

- **Web element locators** (using By)
- **Methods to interact** with those elements (click, send keys, etc.)
- **Assertions** or state-checking methods

Login Test Class: LoginTests.java

LoginTests is a **TestNG test class** that verifies functionalities related to:

1. User login
2. Editing the User's First and Last name (includes scrolling function)
3. Filtering hair care products

Base Test Class: Base_Tests.java

The Base_Tests class serves as a common **base test** class in this Selenium/TestNG framework. Its primary role is to centralize WebDriver setup/teardown and share page-object instances so that individual test classes can inherit and reuse this logic.

The **Setup()** method (annotated with TestNG's @BeforeClass) initializes the browser and navigates to the base URL before any tests in a subclass run.

The **TearDown()** method (annotated with @AfterClass) performs cleanup after all tests in the class have executed. Resetting the user's First and last name, logging out and quitting the driver.

4. Test Execution Flow

Cucumber Sample

1. Cucumber runs the feature file (e.g., login.feature)
2. Step definitions call methods from POM classes (e.g., LoginPage.login())

3. TestRunner runs the feature files implemented

TestNG LoginTests

1.@BeforeClass Setup() method is called

2.Test three scenarios upon the priority given.

2.@TearDown() method is called.